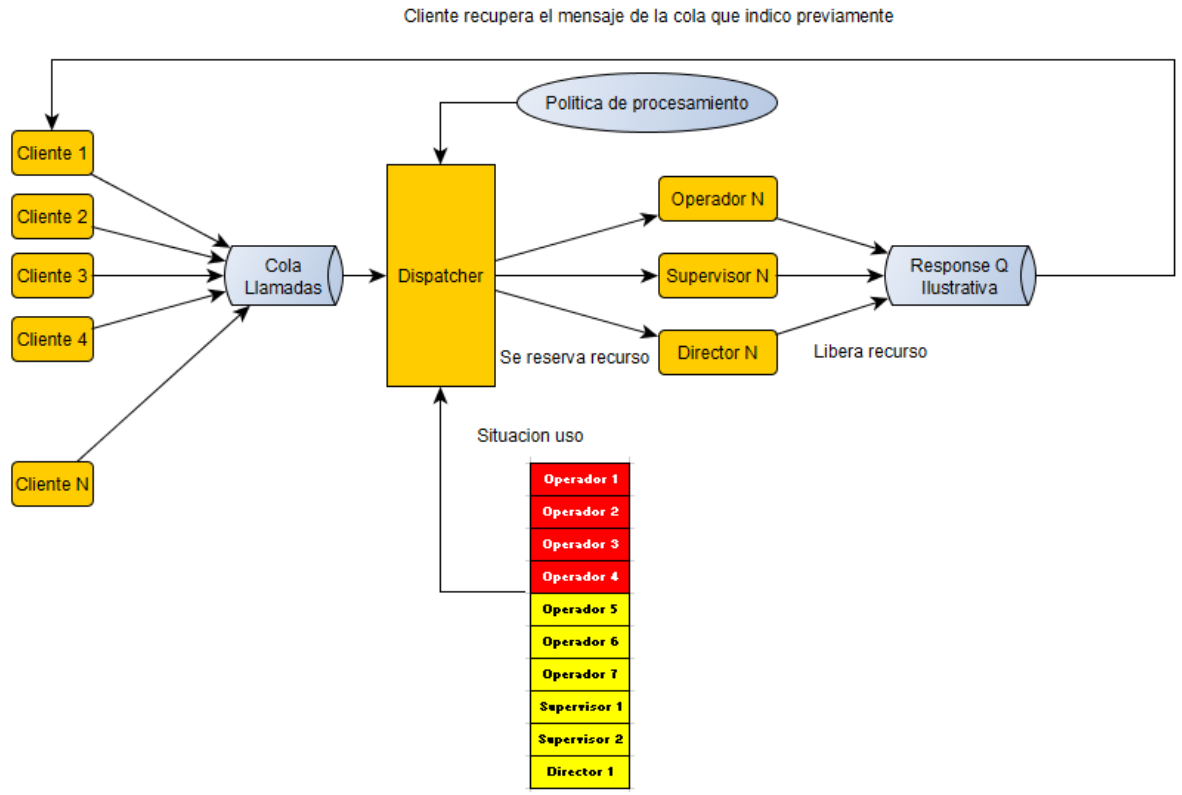
Modelo de atención de llamadas

Descripción de la solución

La solución propuesta se basa en el uso de activeMQ para recibir las llamadas y posteriormente ser procesadas por el dispatcher, de acuerdo a la siguiente figura:



Ante una llamada de un cliente, el sistema recibe la solicitud en la cola de entrada donde el dispatcher, el cual monitorea la cola todo el tiempo, verifica si hay recursos disponibles para procesar la llamada, en caso afirmativo procede a tomar el mensaje de la cola y mandarlo a procesar por el recurso correspondiente, dicho recurso fue previamente seleccionado con el criterio solicitado (primero operadores, luego supervisores y finalmente directores) reservado para procesar y posteriormente cuando el procesamiento haya terminado, marcado para reutilizar.

La respuesta del procesamiento será puesta a disposición del cliente en la cola que haya indicado junto con la llamada. A su vez, la solución implementa una política de procesamiento, Si todos los recursos están en uso y se recibe una nueva llamada, ¿Se descarta o se la pone en espera?; Según la política seleccionada la solución proporcionada permitirá uno de los dos comportamientos (Ver clase PropertiesApp)

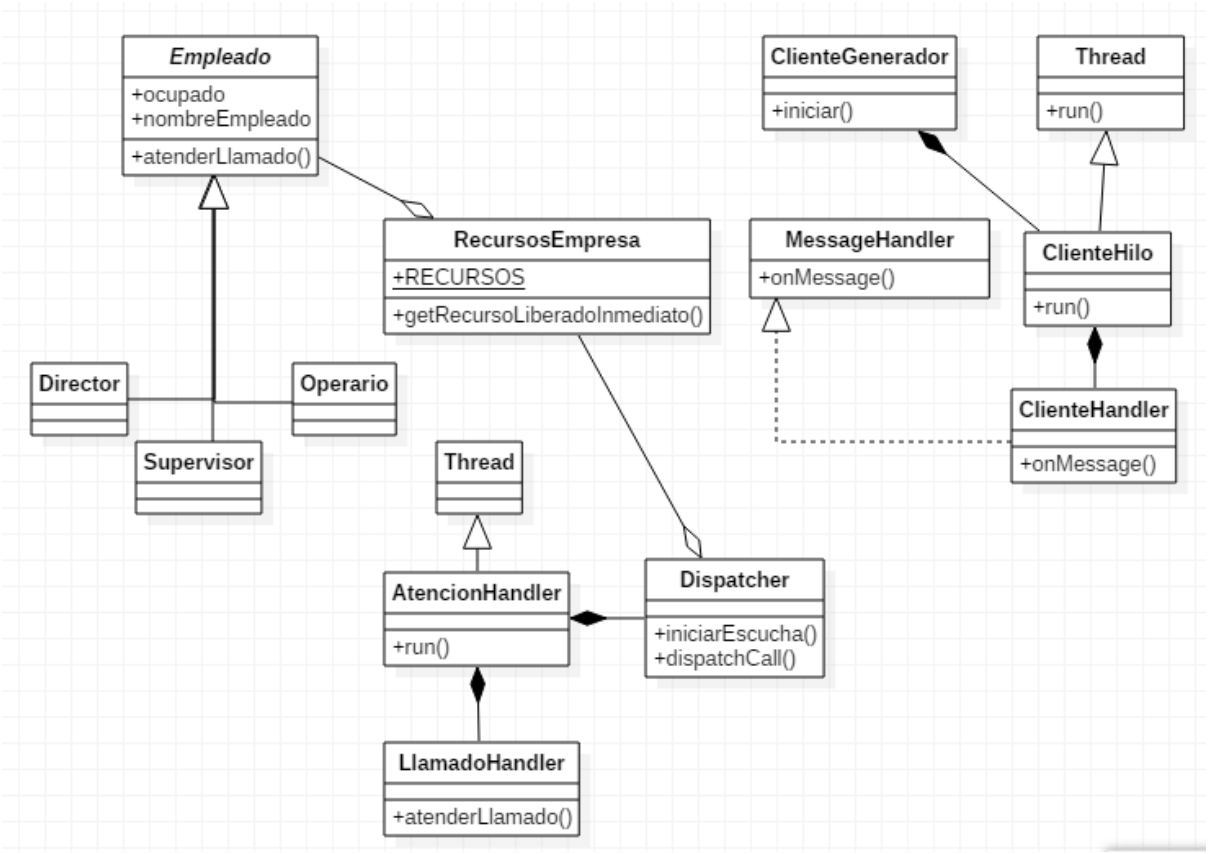
Esta solución permite el procesamiento de mensajes más allá de la cantidad de empleados de la institución, por default tiene configurada la política para colocar las llamadas en espera ante una eventual saturación de recursos.

Arquitectura

Se pueden distinguir tres partes fundamentales a tener en cuenta para puesta en marcha del modelo.

1. Server: Es una instancia de activeMQ que se levanta a través de la clase con el mismo nombre, Un servidor será levantado en el puerto 61616 y quedara a la espera de su utilización. Es el primer proceso que se debe ejecutar.
2. Clientes: Los clientes son disparados empleando hilos puesto que se necesita concurrencia para simular la vida real, La clase ClienteGenerador es la encargada de crear las instancias de los hilos y dispararlas. Si el procesamiento fue correcto, esta clase debería finalizar sin inconvenientes y de forma automática. El parámetro CANTIDAD_CLIENTES indica la cantidad de clientes a ser creados.
3. Dispatcher: Este es el tercer proceso que se debe ejecutar, el mismo levantara el dispatcher que se encargara de (según la política) verificar los recursos disponibles, luego leer una llamada de la cola y disparar el procesamiento en un hilo asincrónico.

Clases del sistema



Consideraciones

Para la solución propuesta se han tenido las siguientes consideraciones:

- Se consideró el uso de Spring para levantar el bróker y los procesos, sin embargo como la mayor parte del código involucra la creación de hilos, lo cual requiere que

se creen con parámetros vía constructor, por lo que la inyección de dependencias de Spring no aporta una mejora importante.

- ¿Por qué JMS? La lectura de la problemática planteada por el ejercicio involucra el manejo de llamadas y su previo análisis para procesamiento, por lo cual se requiere un contenedor previo para encolar los pedidos por lo que JMS se presenta como una alternativa interesante, y particularmente activeMQ debido a que es una implementación liviana y sencilla de levantar. Se pensaron en otros contenedores como una base de datos intermedia con HSQLDB, pero no es su uso natural; por otro lado las estructuras de datos que ofrece Java, tal como Queue, presentaban un modelo demasiado sencillo para modelar el problema.
- Se descartó el uso de un ThreadPoolExecutor fijado a una valor determinado, 10, debido a que no permitía por un lado el control de la selección del recurso a emplear para procesar (llámese operador, supervisor o director) y tampoco la elección de una política de procesamiento, un FixedThreadPool fijado a 10, si recibe un hilo extra, lo almacenará en una cola interna que mantiene el pool, nunca descarta una ejecución.
- Testing, Este tipo de soluciones en la práctica no poseen pruebas del tipo que se realicen con junit debido a que son pruebas de arquitectura basadas en stress y comportamiento más que de lógica y cálculos. Por tal razón para poder probar la solución, se debe recurrir a la ejecución delos procesos especificados anteriormente. En el caso general Jmeter es una herramienta empleada para testear este tipo de soluciones y obtener métricas aplicadas a un entorno en particular.
- El archivo logback.xml tiene la ubicación del archivo de logs, editarlo para ajustar la ubicación deseada.
- Para simplificar el diagrama de clases, se optó por no incluir las clases Utils y los atributos para simplificar la representación. En el código fuente se encuentran los comentarios correspondientes.